

REMARKS

In the Office Action dated April 21, 2004, claims 1-31 were presented for examination. Claims 1-31 were rejected under 35 U.S.C. §102(b) as being anticipated by *Jackson et al.*, U.S. Patent No. 6,473,819.

Applicant wishes to thank the Examiner for the careful and thorough review and action on the merits in this application. The following remarks are provided in support of the pending claims and responsive to the Office Action of April 21, 2004 for the pending application.

In the Office Action of April 21, 2004, the Examiner assigned to the application rejected claims 1-31 under 35 U.S.C. §102(e) as being anticipated by *Jackson et al.* ('819). The *Jackson et al.* patent ('819) relates to a queue lock for a shared memory multiprocessor. More specifically, *Jackson et al.* shows a group of processors coupled to memory via a system bus. The processors are organized across a LAN, but do not share memory. There is no teaching in *Jackson et al.* for organizing any of the processors into a hierarchy. A hierarchy is defined as "an organizational technique in which items are layered or grouped to reduce complexity."¹ In *Jackson et al.* the processors 110a, 110b, and 110c are placed in a local area network (LAN). There is no provision in the LAN of *Jackson et al.* for ordering the processors using their complexity. By definition, such an ordering is required in order to have a hierarchical system. Accordingly, *Jackson et al.* does not teach the hierarchical organization of the processors.

Furthermore, in Applicant's invention the placement of the processor requesting the lock in view of the hierarchical organization of the processors is determinative of processing of the lock. The lock of *Jackson et al.* is responsive to a linear ordering of the processors in the LAN. See Col. 7, lines 50-56. There is no provision in *Jackson et al.* for giving priority for access to a lock to one processor over the other based on an assigned hierarchy. In exceptional cases, *Jackson et al.* uses a linear arrangement of the processors to grant locks based upon a lock-

¹John Lewis and William Loftus, Java Software Solutions: Foundations of Program Design, 669 (3rd ed. 2003), attached as Exhibit A.

granting algorithm to favor the processor to the left or the processor to the right. This is not processing a lock responsive to a hierarchy, it is processing a lock responsive to a linear arrangement. The left or right processor is given priority because of its position in the arrangement; however, its position and the order of the arrangement does not relate to the complexity of the elements. Therefore, the locks of *Jackson et al.* are not responsive to a hierarchy of processors, as the injection of such a hierarchical system into *Jackson et al.* would in fact contradict *Jackson et al.*'s own stated system of assigning locks based upon the linear arrangement of the processors.

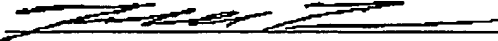
In order for the claimed invention to be anticipated under 35 U.S.C. §102(b), the prior art must teach all claimed limitations presented by the claimed invention. "A claim is anticipated only if each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." MPEP §2131 (citing *Verdegaal Bros. v. Union Oil Co. of California*, 814 F. 2d 628, 631, 2 U.S.P.Q. 2d 1051, 1053 (Fed. Cir. 1987)). As mentioned above, *Jackson et al.* does not show all of the elements as claimed by Applicant in pending claims 1-31. Specifically, *Jackson et al.* does not show a hierarchical grouping of processors, rather *Jackson et al.* merely shows a linear arrangement of processors in a local area network without grouping the processors in a manner which reduces their complexity, *i.e.* a hierarchy. The lock of *Jackson et al.* is processed in response to this linear arrangement of the processors in the network. Applicant's lock is processed with consideration of the hierarchical grouping of the processors. Accordingly, *Jackson et al.* clearly fails to teach the limitations pertaining to the hierarchical grouping of the processors as well the processing of the lock responsive to the hierarchical grouping of the processors as presented in Applicant's pending claims 1-31.

Finally, "[a] previous patent anticipates a purported invention only where, except for insubstantial differences, it contains *all* of the same elements operating in the same fashion to perform an identical function." *Saunders v. Air-Flo Co.*, 646 F.2d 1201, 1203 (7th Cir. 1981) citing *Popeil Brothers, Inc. v. Schick Electric, Inc.*, 494 F. 2d 162, 164 (7th Cir. 1974) (holding patents were not invalid as being anticipated by or obvious in light of prior art) (*emphasis added*). *Jackson et al.* does not anticipate the invention of Applicant based upon the legal

definition of anticipation. Although the prior art cited by the Examiner relates to processors and locks associated therewith, *Jackson et al.* fails to show each and every element as presented in Applicant's claimed invention. In fact, *Jackson et al.* does not show placing the processors into a hierarchy or processing a lock responsive to the hierarchy of the processors. Rather, *Jackson et al.* shows a linear arrangement of processors in a local area network and processing a lock in response to the linear arrangement, which conflicts with each of the Applicant's claimed elements. Accordingly, Applicant respectfully requests the Examiner to remove the rejection of claims 1-31 and to provide allowance of this application.

For the reasons outlined above, withdrawal of the rejection of record and an allowance of this application are respectfully requested.

Respectfully submitted,

By: 
Rochelle Lieberman
Registration No. 39,276
Attorney for Applicant

Lieberman & Brandsdorfer, LLC
12221 McDonald Chapel Drive
Gaithersburg, MD 20878-2252
Phone: (301) 948-7775
Fax: (301) 948-7774
Email: rocky@llegalplanner.com

Date: July 21, 2004

BEST AVAILABLE COPY

APPENDIX A glossary 669

* **hierarchy**—An organizational technique in which items are layered or grouped to reduce complexity.

high-level language—A programming language in which each statement represents many machine-level instructions.

HTML—See HyperText Markup Language.

hybrid object-oriented language—A programming language that can be used to implement a program in a procedural manner or an object-oriented manner, at the programmer's discretion. See also pure object-oriented language.

hypermedia—The concept of hypertext extended to include other media types such as graphics, audio, video, and programs.

hypertext—A document representation that allows a user to easily navigate through it in other than a linear fashion. Links to other parts of the document are embedded at the appropriate places to allow the user to jump from one part of the document to another. See also hypermedia.

HyperText Markup Language (HTML)—The notation used to define Web pages. See also browser, World Wide Web.

icon—A small, fixed-sized picture, often used to decorate a graphical interface. See also image.

identifier—Any name that a programmer makes up to use in a program, such as a class name or variable name.

identity—The designation of an object, which, in Java, is an object's reference name. See also state, behavior.

IEEE 754—A standard for representing floating point values. Used by Java to represent float and double data types.

if—A Java reserved word that specifies a simple conditional construct. See also else.

image—A picture, often specified using a GIF or JPEG format. See also icon.

immutable—The characteristic of something that does not change. For example, the contents of a Java character string are immutable once the string has been defined.

implementation—(1) The process of translating a design into source code. (2) The source code that defines a method, class, abstract data type, or other programming entity.

implements—A Java reserved word that is used in a class declaration to specify that the class implements the methods specified in a particular interface.

import—A Java reserved word that is used to specify the packages and classes that are used in a particular Java source code file.

index—The integer value used to specify a particular element in an array.

index operator—The brackets ([]) in which an array index is specified.

indirect recursion—The process of a method invoking another method, which eventually results in the original method being invoked again. See also direct recursion.

infinite loop—A loop that does not terminate because the condition controlling the loop never becomes false.

infinite recursion—A recursive series of invocations that does not terminate because the base case is never reached.

infix expression—An expression in which the operators are positioned between the operands on which they work. See also postfix expression.

inheritance—The ability to derive a new class from an existing one. Inherited variables and methods of the original (parent) class are available in the new (child) class as if they were declared locally.

initialize—To give an initial value to a variable.

initializer list—A comma-separated list of values, delimited by braces {}, used to initialize and specify the size of an array.

inline documentation—Comments that are included in the source code of a program.

inner class—A nonstatic, nested class.

input/output buffer—A storage location for data on its way from the user to the computer (input buffer) or from the computer to the user (output buffer).

EXHIBIT

A